1984

On Terminal Assignments That Minimize the Density

Mikhail J. Atallah
Purdue University, mja@cs.purdue.edu

Susanne E. Hambrusch
Purdue University, seh@cs.purdue.edu

Report Number:
84-468

On Terminal Assignments That Minimize the Density

Mikhail J. Atallah
Susanne E. Hambrusch

Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

CSD-TR-468 February 1984

Abstract

Terminal assignment problems are placement problems in channel routing in which each one of n entry terminals on one side of the channel is assigned to one of m exit terminals on the other side of the channel. We study solutions to terminal assignment problems that minimize the density, a cost measure closely related to the minimum channel width needed to wire the resulting channel routing problem. We present a new characterization of the optimal achievable density that leads to efficient algorithms for a number of terminal assignment problems, and we show that other assignment problems are NP-hard.

Key Words

Analysis of algorithms, channel routing, density, binary search, NP-hard problems.

1. Introduction

Channel routing, the problem of connecting terminals across a channel, arises in many layout systems for integrated circuits. The decisions of where to place the terminals and how to select the terminals to be connected are made before the routing and have considerable influence on the cost measures of the routing, such as the channel width. In this paper we consider problems in which an optimal assignment of every terminal on the upper row to a terminal on the lower row of the channel is determined so that given conditions are satisfied and the routing step produces a channel of minimum width. For a number of such assignment problems we present efficient algorithms, and others we show to be NP-hard.

We are given the positions of n entry terminals, $p_1, \ldots, p_n$, $p_1 < \cdots < p_n$, on the upper row of the channel, and the positions of m exit terminals, $q_1, \cdots, q_m$, $q_1 < \cdots < q_m$, on the lower row of the channel, $1 \le p_i, q_i \le M$, $m \ge n$. Two of the problems discussed in this paper are the one-to-any problem and the choice problem. In the *one-to-any problem* we have to compute a one-to-one function $f : [1{:}n] \to [1{:}m]$ so that every entry terminal p_i is assigned to the exit terminal $q_{f(i)}$. Furthermore, the channel routing problem (CRP) consisting of the pairs, called *nets*, $(p_1, q_{f(1)}), \cdots, (p_n, q_{f(n)})$ must have minimum density over all assignments. The *density* of the CRP is the maximum over all x of the number of nets $(p_i, q_{f(i)})$ for which $p_i \le x < q_{f(i)}$ or $p_i > x \ge q_{f(i)}$. The density is closely related to the minimum channel width, and minimizing the density also minimizes the channel width in routing models that consist of at least 2 layers [BB, PL, RBM]. The *choice problem* is similar to the one-to-any problem, except that each entry terminal p_i has a set P_i of exit terminals associated with it, and we require that $q_{f(i)} \in P_i$ for every i. One of the results of this paper is that the one-to-any problem can be solved in $O(n+m)$ time, while the choice problem is NP-hard.

Both problems, the one-to-any and the choice problem, can be viewed as bipartite matching problems in which a quantity, namely the density, has to be minimized. But none of the known bipartite matching algorithms [L] provide any insight into matchings that minimize the density. In this paper we develop a technique that solves a number of problems, including the one-to-any problem, by a factor of $\log n$ faster over the solutions which use previously known techniques. We also show that more general assignment problems, such as the choice problem, are NP-hard. Thus, the complexity of matching problems which minimize the density differs considerably from the complexity of the standard bipartite matching problems.

The main contribution of this paper with respect to efficient algorithms is a new combinatorial characterization of the best achievable density. This characterization allows us to compute the best achievable density for assignment problems efficiently. The actual assignment achieving this optimal density is determined once this density is known. Using this approach we solve the one-to-any problem in time $O(n+m)$, compared to time $O((n+m)\log n)$ achieved by previously known techniques. We also consider the one-to-any problem with *fixed nets* in which the input includes h additional nets, each of which has its entry and exit terminals already determined. Thus, even before the assignment is done, some positions in the channel have a non-zero density because of the fixed nets. We show (i) how to compute the best achievable density for the one-to-any problem with h fixed nets in time $O(n+m+h)$, and (ii) how to obtain an assignment achieving this density in time $O((n+m+h)\log(n+h))$. In both cases we improve results obtained by known techniques by a factor of $\log n$.

A natural extension of the one-to-any problem is the *range problem*. Each entry terminal p_i now has a range $[l_i, r_i]$ associated with it, and the assigned exit terminal $q_{f(i)}$ has to be in this range. If $l_1 \leq l_2 \leq \cdots \leq l_n$ and

$r_1 \leq r_2 \leq \cdots \leq r_n$, the problem is called the *ordered range problem*. We show how to generate the optimal assignment for this problem in $O(n+m)$ time. We also consider the ordered range problem when fixed nets are present.

The paper is organized as follows. In Section 2 we consider the one-to-any problem, and we discuss the main ideas used for pre-computing the best achievable density. Section 3 shows how this technique can be extended to the one-to-any problem with fixed nets. In Section 4 we solve the ordered range problem with or without fixed nets efficiently. Section 5 contains the NP-hardness results: We show that the choice problem and the weighted version of the one-to-any problem are NP-hard. In Section 6 and 7 we present efficient solutions for two other terminal assignment problems. The first of these is range problem when every $l_i = 1$, and $m = M$, for which we we give an $O(n \log n)$ time algorithm. The second of these is the version of the previous problems in which only s of the n entry terminals have to be assigned to an exit terminal.

Different placement problems for channel routing have been discussed and analyzed in recent papers. The offset problem [DKSSU, LaP, M] and the rotation problem [AH] can be solved efficiently on a number of routing models. Other efficiently solvable placement problems can be found in [LeP, GCW]. Inherently difficult placement problems for channel routing are discussed in [GCW, LL, P].

2. The One-to-Any Problem

Throughout this paper the input includes two lists. One list contains n entry terminals $p_1, \ldots, p_n$, $1 \leq p_1 < \cdots < p_n \leq M$, and the second list contains m exit terminals $q_1, \ldots, q_m$, $1 \leq q_1 < \cdots < q_m \leq M$, $m \geq n$. In the *one-to-any problem* we compute a one-to-one function $f : [1,n] \rightarrow [1,m]$ such that if every p_i is assigned to $q_{f(i)}$, then the CRP consisting of the nets $(p_i, q_{f(i)})$ has

minimum density. We use p_i (q_i) to denote both the terminal p_i (q_i) and the value of its abscissa (which is an integer between 1 and M). Hence, $q_j < p_i$ means that exit terminal q_j is to the left of entry terminal p_i. We will furthermore assume that $p_i \neq q_j$ for every i and j. This assumption is made only for convenience, because if $p_i = q_j$, then we can assign q_j to p_i (i.e., set $f(i)=j$) and ignore the net.

If the solution contains p_i and p_j with $p_i < p_j$ and $q_{f(i)} > q_{f(j)}$, then we can always 'uncross' by assigning the exit terminal $q_{f(j)}$ to p_i and $q_{f(i)}$ to p_j. Doing this cannot increase the density. Thus, there exists an optimal solution with $q_{f(1)} < q_{f(2)} < \cdots < q_{f(n)}$. Therefore the crucial step of the algorithm is to select n of the m exit terminals which, when assigned to the entry terminals, result in a CRP of minimum density. See Fig. 2.0 for an non-optimal and an optimal selection.

Given an integer d, $d \leq n$, we can determine in time $O(n+m)$ whether or not there exists an assignment whose density is $\leq d$. The algorithm for doing this simultaneously scans, from left to right, the list of the entry terminals and that of the exit terminals. Throughout the paper, we use "assigning p_i to q_j" and "joining p_i to q_j" interchangeably. Assume we are at p_i and q_j in our scan of the list of entry and exit terminals, respectively. We assign q_j to p_i only when doing so does not cause the density to exceed d. If we join p_i and q_j, then we continue with p_{i+1} and q_{j+1}, otherwise we continue with p_i and q_{j+1}. The scan terminates with success as soon as it joins p_n, and with failure if it fails to join some p_i to q_m (i.e., it 'runs out' of exit terminals). The fact that the scan is at p_i and q_j implies that each of $p_1, \ldots, p_{i-1}$ already has an exit terminal assigned to it, and that $q_j > q_{f(i-1)}$. We note an interesting property of the nets produced by this strategy: If p_i gets joined to q_j and $p_i < q_j$, then *all* of the exit terminals between p_i and q_j have been joined to entry terminals to the left of p_i; i.e., none of them is available. We call this the *left-*

compressed property. For example, the solution of Figure 2.0(b) is not left-compressed (it would be left-compressed if p_3 was joined to q_3 instead of q_4).

We now briefly outline how to determine whether joining p_i to q_j causes the density to exceed d. Consider first the case when $q_j < p_i$. Let p_k be the leftmost entry terminal with $p_k > q_j$. We can join p_i and q_j without causing the density to exceed d if and only if $i-k<d$. Now consider the case when $q_j > p_i$. Let p_k be the leftmost entry terminal with $q_{f(k)} > p_i$. We can join p_i and q_j if and only if $i-k<d$. The reader may notice that if $i-k=d$, then we already know that the algorithm will stop with failure.

The proof of correctness of the greedy verification algorithm described above is left to the reader. The algorithm can clearly be implemented to run in time $O(n+m)$, and it suggests an $O((n+m)\log\hat{d})$ time algorithm for determining the optimal density $\hat{d}$ by binary search. Such a binary search technique has been used in [LeP] for River Routing problems. The main result of this section is to give a new combinatorial characterization of the optimal density, resulting in an $O(m+n)$ time algorithm for computing it. Once we have the optimal density, the $O(m+n)$ time verification algorithm outlined above can be used to produce an optimal assignment of exit to entry terminals.

Throughout the paper, a net which crosses from position x to position $x+1$ will be said to *cross column* x. More formally, a net $(p_i, q_{f(i)})$ crosses column x if $p_i \le x < q_{f(i)}$ or $q_{f(i)} \le x < p_i$.

Definition 2.1 For all x and y, $1 \le x \le y \le M$, we define $up(x,y)$ (resp. $down(x,y)$) to be the number of entry (resp. exit) terminals whose position is $\ge x$ and $\le y$. Let $out(x,y) = up(x,y) - down(x,y)$.

We now define a quantity Φ which, we will later show, equals the optimal density $\hat{d}$.

Definition 2.2 Let Φ be defined as follows:

$$\Phi = Max\{ \max_{x,y} \lceil out(x,y)/2 \rceil , \max_{x} out(1,x) , \max_{x} out(x,M) \}$$

Note that $out(p_i,p_i)=1$, and therefore $\Phi \geq 1$.

Lemma 2.3 $\hat{d} \geq \Phi$.

Proof: For every x and y, $1 \leq x \leq y \leq M$, at least $out(x,y)$ nets must cross columns $x-1$ and column y. Therefore, at least one of these columns is crossed by $\lceil out(x,y)/2 \rceil$ nets. This implies that $\hat{d} \geq \lceil out(x,y)/2 \rceil$. For every x, $1 \leq x \leq M$, at least $out(1,x)$ (resp. $out(x,M)$) wires cross column x (resp. column $x-1$), which implies that $\hat{d} \geq out(1,x)$ (resp. $\geq out(x,M)$). □

We now show that Φ can be computed in time $O(m+n)$. In order to do so, we need the following Lemma.

Lemma 2.4 Given a sequence of positive and negative numbers $c_1, \cdots, c_n$, it is possible to find, in time $O(n)$, indices i and j, $i \leq j$, such that $c_i + c_{i+1} \cdots + c_j$ is largest over all possible values i and j.

Proof: The proof is easy and is omitted. □

Lemma 2.5 Φ can be computed in time $O(n+m)$.

Proof: It is clear that computing each of $\max_{x} out(1,x)$ and $\max_{x} out(x,M)$ can be done in time $O(m+n)$. Note that $\max_{x,y} \lceil out(x,y)/2 \rceil$ is equal to $\left\lceil \frac{1}{2} \max_{x,y} out(x,y) \right\rceil$. Computing $\max_{x,y} out(x,y)$ can be done in time $O(n+m)$ by (i) merging the list of the p_i's and that of the q_i's and, (ii) in the resulting sorted sequence $a_1, \cdots, a_{m+n}$, replacing every a_i by $+1$ if $a_i = p_j$ and by -1 if $a_i = q_j$, then (iii) in the resulting (not sorted) sequence $c_1, \cdots, c_{m+n}$ of +1's and -1's, finding i and j, $i \leq j$, for which $c_i + \cdots + c_j$ is largest. $\max_{x,y} out(x,y)$ is the sum $c_i + \cdots + c_j$. Steps (i) and (ii) can clearly be done in time $O(m+n)$. Step (iii) takes time $O(m+n)$ as a consequence of Lemma 2.4. □

We now show that a solution which achieves density equal to Φ exists.

Lemma 2.6 $\hat{d} = \Phi$.

Proof: Lemma 2.3 implies that it suffices to show that $\hat{d} \leq \Phi$. To do this, we use the *assigning strategy* described in the verification algorithm: Simultaneously scan the list of the entry terminals and that of the exit terminals, joining the currently scanned entry terminal to the next available exit terminal that does not cause the density to exceed Φ. (An available exit terminal is one that has not yet been assigned to a p_i.) If the assigning strategy terminates with failure then one of the two following situations must have occurred.

A. An entry terminal p_{n-i} cannot be joined to q_{m-i} and $q_{m-i} < p_{n-i}$. Note that even if the assigning strategy joins p_{n-i} to an exit terminal to the right of q_{m-i}, we eventually run out of exit terminals.

B. Some p_k cannot be joined to the next available exit terminal q_s, and $p_k < q_s$.

From now on, we assume that the assigning strategy stops with failure as soon as either situation A or B occurs. We claim that situation A or B cannot occur; i.e., we succeed in joining every entry terminal to an exit terminal without exceeding density Φ. We prove the claim by contradiction. We obtain the desired contradiction by examining in detail what prevented us from joining p_{n-i} to q_{m-i} in situation A, and what prevented us from joining p_k to q_s in situation B.

Situation A. In this situation we were unable to join p_{n-i} to q_{m-i} because there are Φ entry terminals between q_{m-i} and p_{n-i}, all of which were joined to exit terminals to the left of q_{m-i}. But then, we have $out(q_{m-i}+1, M) = \Phi + i + 1 - i = \Phi + 1$, a contradiction. See Figure 2.1.

Situation B. We distinguish two cases.

Case 1: There are no available exit terminals to the left of q_s. See Figure 2.2.

In this case we were unable to join p_k to q_s because there are Φ exit terminals between p_k and q_s, all of which were joined to entry terminals to the left of p_k. But then, $out(1,p_k)=\Phi+1$, a contradiction.

Case 2: There is (one or more) available terminal(s) to the left of q_s. See Figure 2.3. Let q_j be the rightmost available exit terminal that is to the left of q_s. Note that q_j is to the left of p_k (because of the left-compressed property). Then there are Φ entry terminals immediately to the right of q_j, all of which were joined to exit terminals to the left of q_j (otherwise q_j would not be available). But then, we have $out(q_j+1,p_k)=2\Phi+1$, which in turn implies that

$$\left\lceil out(q_j+1,p_k)/2 \right\rceil = \Phi+1,$$

another contradiction. □

We are now ready to state the main result of this Section.

Theorem 2.7 The one-to-any problem can be solved in time $O(n+m)$.

Proof: The algorithm and its correctness follow from the preceding discussion. □

3. The One-to-Any Problem With Fixed Nets

In this Section we consider the one-to any problem as defined in Section 2 when, in addition to the p_i's and q_i's, we have h fixed nets. A *fixed net* consists of an entry and an exit terminal that are already joined. The fixed nets are given as a list of h entry terminals $v_1, \ldots, v_h$, $v_1 < \cdots < v_h$, a list of h exit terminals $w_1, \ldots, w_h$, $w_1 < \cdots < w_h$, and an h-element permutation g such that every $(v_i, w_{g(i)})$ is a net, $1 \le i \le h$. The problem is to assign n of the q_i's to the p_i's so that the resulting CRP has minimum density. The resulting CRP now consists of the nets $(p_i, q_{f(i)})$, $1 \le i \le n$, and the fixed nets $(v_i, w_{g(i)})$, $1 \le i \le h$. See Figure 3.0 for an example. We first show how to compute the

optimal density of a one-to-any problem with h fixed nets in time $O(n+m+h)$.

As in Section 2, there exists a solution of minimum density in which $q_{f(1)} < \cdots < q_{f(n)}$, and the problem is again that of selecting n of the m exit terminals that will be joined to the p_i's. We define $up(x,y)$ ($down(x,y)$) as the number of p_i's (q_i's) that are $\geq x$ and $\leq y$, and let $out(x,y)$ denote the quantity $up(x,y)-down(x,y)$. Note that the functions *up*, *down* and *out* do not depend on the fixed nets. The contribution of the fixed nets to the density of column x is expressed in $cross(x)$, which equals the number of fixed nets crossing column x. Recall that a net $(v_i, w_{g(i)})$ crosses column x if and only if $v_i \leq x < w_{g(i)}$ or $w_{g(i)} \leq x < v_i$. We now define a quantity Ψ which we will later show to be equal to the optimal density $\hat{d}$.

Definition 3.1 Let Ψ be defined as follows:

$$\Psi = Max \{ \; \max_{x,y} \lceil (cross(x-1)+cross(y)+out(x,y))/2 \rceil, \; \max_{x} (cross(x)+out(1,x)), \; \max_{x} (cross(x-1)+out(x,M)) \; \}$$

Lemma 3.2 $\hat{d} \geq \Psi$.

Proof: The proof is along the lines of that of Lemma 2.3 and is omitted. □

Lemma 3.3 Ψ can be computed in time $O(m+n+h)$.

Proof: The proof is along the lines of that of Lemma 2.5, and is therefore omitted. □

Lemma 3.4 $\hat{d} = \Psi$.

Proof: Lemma 3.2 implies that it suffices to show that $\hat{d} \leq \Psi$; i.e., density Ψ can be achieved. We use the same assigning strategy as in the proof of Lemma 2.6, except that now Ψ plays the role of Φ. The observations made about the assigning strategy at the beginning of the proof of Lemma 2.6 still hold. We obtain contradictions by examining why we could not join p_{n-i} to

q_{m-i} in situation A, or p_k to q_s in situation B.

Situation A. In this situation we cannot join p_{n-i} to q_{m-i} because there exists a column x, $q_{m-i} \leq x < p_{n-i}$, in which the density is already Ψ. Then there are $\alpha = \Psi - cross(x)$ p_j's at positions $> x$ and $< p_{n-i}$ that are joined to q_j's that are $< q_{m-i}$. This implies that $cross(x) + out(x+1, M) = cross(x) + \alpha + i + 1 - i = \Psi + 1$, a contradiction.

Situation B. We distinguish two cases.

Case 1: There are no available exit terminals to the left of q_s. In this case we cannot join p_k to q_s because there exists a column x, $p_k \leq x < q_s$, in which the density is already Ψ. Then there are $\alpha = \Psi - cross(x)$ q_j's at positions $> x$ and $< q_s$ that are joined to entry terminals to the left of p_k. But then, we have $cross(x) + out(1, x) = cross(x) + \alpha + 1 = \Psi + 1$, a contradiction.

Case 2: There is (one or more) available terminal(s) to the left of q_s. Let q_j be the rightmost available exit terminal that is to the left of q_s. See Figure 3.1. Note that q_j is to the left of p_k because of the left-compressed property. Then the following holds:

(i) There exists a column y, $p_k \leq y < q_s$, such that column y has currently density Ψ. Then there are $\Psi - cross(y)$ q_r's at positions $> y$ and $< q_s$ that are joined to p_r's to the left of p_k. This follows from the fact that we are unable to join p_k to q_s.

(ii) There exists a column x, $q_j \leq x < p_k$, in which the density is Ψ. Then $\Psi - cross(x)$ p_r's at positions $> x$ are joined to exit terminals to the left of q_j. If such an x did not exist, then q_j would not be available.

Then (i) and (ii) imply that

$$\lceil (out(x+1,y)+cross(x)+cross(y))/2 \rceil \geq \lceil (\Psi+\Psi+1)/2 \rceil = \Psi+1,$$

a contradiction. □

Theorem 3.5 The optimal density of a one-to-any problem with h fixed nets can be computed in time $O(n+m+h)$.

Proof: Follows from the last two Lemmas. □

We next consider the implementation of the algorithm that produces the assignment achieving density Ψ. The idea of the algorithm is the same as in Section 2: Scan the list of the p_i's and that of the q_j's from left to right and assign p_i to q_j if density Ψ is not exceeded. When no fixed nets were present, only one column had to be checked in order to determine whether or not the density is exceeded. Now that fixed nets are present, density Ψ could be exceeded in any column between p_i and q_j. We outline an algorithm that produces the assignment in time $O((n+m+h)\log(n+h))$.

During the scan of the lists of entry and exit terminals we use a modified 2-3 tree structure T in which the leaves contain entry and exit terminal positions in ascending order. Initially, only the entry and exit terminals of the fixed nets are stored in T. The nodes of T have additional entries that allow us to compute the current maximum density between two positions in the channel in $O(\log(n+h))$ time. Assume we are at p_i and q_j in the scan. We then use T to determine the current maximum density between p_i and q_j. If it is less than Ψ, we insert the net $(Max(p_i,q_j), Min(p_i,q_j))$ into T, and continue with p_{i+1} and q_{j+1}. If joining p_i to q_j exceeds density Ψ, we continue with p_i and q_{j+1}. Inserting a net into T and recording its contribution to the density in the columns between p_i and q_j can also be done in $O(\log(n+h))$ time. The implementation details of the assignment algorithm outlined above are left to the reader. The rebalancing procedure for the tree structure is similar to the one described in [AHU].

Theorem 3.6 An assignment to the one-to-any problem with h fixed nets achieving density Ψ can be obtained in $O((n+m+h)\log(n+h))$ time.

Proof: The optimal density Ψ can be computed in $O(n+m+h)$ time by Theorem 3.5. The tree structure T described above allows us to obtain the n exit terminals to be assigned to the n entry terminals in time $O((n+m+h)\log(n+h))$. □

4. Ranges and Fixed Nets

A more general version of the one-to-any problem with (or without) fixed nets is the *ordered range problem* with (or without) fixed nets. In the ordered range problem we are given for every entry terminal p_i two positions l_i and r_i with $l_i \le r_i$, $1 \le i \le n$, and we have $l_1 \le l_2 \le \cdots \le l_n$ and $r_1 \le r_2 \le \cdots \le r_n$. Each p_i has to be assigned to an exit terminal $q_{f(i)}$ that is in the range $[l_i, r_i]$; i.e., $l_i \le q_{f(i)} \le r_i$. We say that q_j is *in the range of* p_i if and only if $l_i \le q_j \le r_i$. The reader can easily verify that in the ordered range problem there exists an optimal solution with $q_{f(1)} < \cdots < q_{f(n)}$. Fixed nets are represented as in Section 3; i.e., the net $(v_i, w_{g(i)})$, $1 \le i \le h$, is a fixed net. We first show that the optimal density of the ordered range problem can again be computed in $O(n+m+h)$ time.

We assume for the time being that the l_i's (resp. r_i's) are distinct. We will later show that any problem for which this assumption is not satisfied can be reduced to an equivalent problem where it is satisfied. We furthermore assume that every l_i (resp. r_i) coincides with a q_j, and that every q_j is in the range of a p_i. (If it is not, it is useless and can be removed.)

In addition to the functions *out*, *up*, *down*, and *cross* defined in Section 3, we define two new functions, *inl* and *inr*: *inl*(x) (resp. *inr*(x)) is the number of non-fixed nets that have to cross column x because their p_i is $\le x$ (resp. $> x$) while their $l_i > x$ (resp. $r_i \le x$). Observe that in the ordered range

problem at least one of $inl(x)$ and $inr(x)$ is zero.

Definition 4.1 Let Λ be defined as follows:

$$\Lambda = Max \{ \quad \max_{x,y} \lceil (cross(x-1)+cross(y)+out(x,y))/2 \rceil, \quad \max_{x} (cross(x)+inl(x)+inr(x)) \ \}$$

Note that in the quantities Φ and Ψ defined in Sections 2 and 3, respectively, the second and third term were not combined for the sake of clarity. If $\hat{d}$ denotes the optimum density we have the following Lemma.

Lemma 4.2 $\hat{d} \geq \Lambda$.

Proof: Once it is observed that $inl(x).inr(x)=0$, the proof becomes similar to that of Lemma 3.2, and is therefore omitted. □

Lemma 4.3 Λ can be computed in time $O(m+n+h)$.

Proof: The proof is along the lines of that of Lemma 2.5, and is therefore omitted. □

Lemma 4.4 $\hat{d} = \Lambda$.

Proof: Lemma 4.2 implies that it suffices to show that density Λ can be achieved. We modify the assigning strategy described in the proof of Lemma 2.6 to take care of the fact that p_i cannot be joined to any q_j that is not in its range. The assigning strategy stops with failure if some p_i cannot be joined to an exit terminal in its range. We now prove that this assigning strategy must succeed. Suppose that, to the contrary, we fail to join p_i to any of the q_j's in its range. We distinguish two cases.

Case 1: We were unable to join p_i to r_l, $r_l < p_i$. Note that r_l must be available because it is not in the range of any of the p_j's that are to the left of p_i (recall that we are temporarily assuming that the r_j's are distinct). In this case there must be some x, $r_i \leq x < p_i$, such that there are $\Lambda - cross(x)$ p_j's that are $\geq x$ and $< p_i$ and are joined to q_j's that are $< r_i$. (If there were no such x then we would be able to

join p_i and r_i.) Those $\Lambda - cross(x)$ p_j's are to the left of p_i and hence their r_j's are $< r_i$. This implies that $inr(x) \geq \Lambda - cross(x)+1$, and therefore $inr(x)+cross(x) \geq \Lambda+1$, a contradiction.

Case 2: We were unable to join p_i to r_i, $r_i > p_i$. Let q_α be the position of the leftmost available exit terminal that is in the range of p_i with $q_\alpha > p_i$ (possibly, $q_\alpha = r_i$). Then there must exist some y, $p_i \leq y < q_\alpha$, such that there are $\Lambda - cross(y)$ unavailable q_j's that are $> y$ and $< q_\alpha$ (otherwise p_i would have been joined to q_α). Those $\Lambda - cross(y)$ q_j's are joined to the $\Lambda - cross(y)$ p_k's that are immediately to the left of p_i. Now, we distinguish two sub-cases.

Sub-case A: All the q_j's to the left of q_α are unavailable. In this case $q_\alpha = l_i$ and every p_k to the left of p_i is joined to its l_k. But then, $inl(y) \geq \Lambda - cross(y)+1$, which implies that $inl(y)+cross(y) \geq \Lambda+1$, a contradiction.

Sub-case B: There exists at least one available exit terminal to the left of q_α. Let q_β be the rightmost available exit terminal to the left of q_α. There must be some x between q_β and p_i such that $\Lambda - cross(x-1)$ p_k's with $p_k \geq x$ are joined to q_j's that are $< x$. Note that such an x must exist both when $q_\beta > l_i$ and when $q_\beta \leq l_i$, since otherwise q_β would not be available (because it is in the range of at least one p_j). But then, we have

$$\lceil (out(x,y)+cross(x-1)+cross(y))/2 \rceil = \lceil (2\Lambda+1)/2 \rceil = \Lambda+1,$$

a contradiction. □

We now show that an ordered range problem which does not satisfy the assumptions we made earlier can be transformed in $O(n+m)$ time into an equivalent problem which does satisfy these assumptions. Recall that these assumptions were that the l_i's are distinct, the r_i's are distinct, that each l_i

and r_i coincides with a q_j, and that every q_j is in the range of at least one p_i. First, the problem is modified in an obvious way so that the last two assumptions hold. Then, the l_i's are made distinct as follows: If p_i and p_{i+1} have $l_i = l_{i+1} = q_j$, for some j, we change the value of l_{i+1} to q_{j+1}. Such a change is justified by the facts that (i) no more than one entry terminal can get assigned to position l_i, and (ii) there is an optimal solution without any crossings. The r_i's are made distinct in a symmetric manner. We leave it to the reader to verify that the above transformations can be implemented in $O(n+m)$ time by scanning the list of p_i's and that of the q_i's first from left to right and then from right to left. This scan is also used to detect whether or not the given range problem has a solution, i.e. whether it is possible at all to assign to every p_i a q_j without violating the range constraints.

Theorem 4.5 The optimal density of an ordered range problem with h fixed nets can be computed in time $O(n+m+h)$.
Proof: Follows from the above Lemmas.

Theorem 4.6 An assignment to the ordered range problem achieving density Λ can be obtained in $O(n+m)$ time when no fixed nets are given.
Proof: The algorithm is similar to the assignment algorithm described in Section 2, and details are omitted. □

Theorem 4.7 An assignment to the ordered range problem achieving density Λ can be obtained in time $O((n+m+h)\log(n+m))$ when h fixed nets are given.
Proof: Similar to the proof of Theorem 3.6. □

5. Generalizations which are NP-hard

In this Section we show that two generalizations of the assignment problems discussed in the previous sections are NP-hard. Consider first the

weighted one-to-any problem: Assume a positive weight w_i is associated with every p_i. The density of column x is now defined to be the sum of the weights of the nets that cross column x. For example, if the w_i's of the CRP shown in Figure 2.0(a) are 15,7,12, then the density is 19.

Theorem 5.1 The weighted version of the one-to-any problem is NP-hard.

Proof: The proof is by a transformation from PARTITION. Let $a_1, \ldots, a_n$ be an instance of the PARTITION problem [GJ], and create in polynomial time the following instance of the weighted one-to-any problem. Let $M=3n+2$, and create $n+2$ entry terminals $p_0, \cdots, p_{n+1}$ such that $p_i=n+i+1$, $0 \le i \le n+1$. Also create $2n$ exit terminals $q_1, \cdots, q_{2n}$ such that $q_i=i$ if $1 \le i \le n$, and $q_i=n+2+i$ if $n+1 \le i \le 2n$. (See Figure 5.1.) In addition, if we let w_i denote the weight associated with p_i, then set $w_0=w_{n+1}=a_1+\cdots+a_n$, and set $w_i=a_i$ for $1 \le i \le n$. The minimum density in the resulting weighted one-to-any problem equals $3(a_1+\cdots+a_n)/2$ if and only if the PARTITION problem has a solution. □

The second problem shown to be NP-hard is the most general terminal assignment problem, the *choice problem*. In the choice problem we are given for every entry terminal p_i a set P_i of m_i exit terminals. We have to assign each entry terminal p_i to one of the exit terminals in P_i (i.e., $q_{f(i)} \in P_i$) so that the CRP consisting of the nets $(p_i, q_{f(i)})$ has minimum density. The P_i's need not be disjoint. We show that the choice problem with every $m_i=2$, called the 2-choice problem, is already NP-hard. The proof is by a transformation from monotone 3-SAT, in which every clause of a boolean formula in conjunctive normal form contains either 3 unnegated or 3 negated variables, [GJ].

Theorem 5.2 The 2-choice problem is NP-hard.

Proof: Let $C_1, C_2, \cdots, C_{k_1}$ be the k_1 clauses which contain only unnegated

variables, and which we call the positive clauses. Let $D_1, D_2, \cdots, D_{k_2}$ be the k_2 clauses which contain only negated variables, and which we call the negative clauses. Let $x_1, x_2, \cdots, x_n$ be the variables used in the k_1+k_2 clauses, and let o_i (resp. $\bar{o}_i$) be the number of occurrences of x_i in the positive (resp. negative) clauses. The l-th leftmost occurrence of the variable x_i in the positive (resp. negative) clauses is referred to as x_{il} (resp. $\bar{x}_{il}$), $1 \le i \le n$, $1 \le l \le o_i$ (resp. $1 \le l \le \bar{o}_i$). We can assume that each variable x_i occurs in at least one positive clause and in at least one negative clause. If x_i appears only in positive (resp. negative) clauses we can set x_i to true (resp. false) and delete all the clauses containing x_i (resp. $\bar{x}_i$).

Given an instance of the monotone 3-SAT problem we create the following instance of the choice problem on a channel of length $M = 6(k_1+k_2)+2n$. For each clause we create 6 entry and 6 exit terminals. To the left of the $6(k_1+k_2)$ terminals associated with the clauses we create n entry terminals at positions $1, \cdots, n$, and to the right we create n exit terminals at positions $n+6(k_1+k_2)+1, \cdots, 2n+6(k_1+k_2)$. See Figure 5.2. For each positive clause $C_r = (x_{il_1} \vee x_{jl_2} \vee x_{kl_3})$ the 6 entry terminals are called x_{il_1}, x_{jl_2}, x_{kl_3}, y_{il_1}, y_{jl_2}, and y_{kl_3}. Below these are 6 exit terminals: u_{il_1}, u_{jl_2}, u_{kl_3}, $\bar{u}_{il_1}$, $\bar{u}_{jl_2}$, and $\bar{u}_{kl_3}$. See Figure 5.3(a). The entry terminal x_{il_1} and the exit terminal u_{il_1} are at position $n+6(r-1)+1$. For each negative clause $D_r = (\bar{x}_{il_1} \vee \bar{x}_{jl_2} \vee \bar{x}_{kl_3})$ the 6 entry terminals are $\bar{x}_{il_1}$, $\bar{x}_{jl_2}$, $\bar{x}_{kl_3}$, $\bar{y}_{i_{l_1}}$, $\bar{y}_{jl_2}$, and $\bar{y}_{kl_3}$, and the 6 exit terminals are $\bar{v}_{il_1}$, $\bar{v}_{jl_2}$, $\bar{v}_{kl_3}$, v_{il_1}, v_{jl_2}, and v_{kl_3}. See Figure 5.3(b). Setting x_i = true will correspond to joining every x_{il} to u_{il}, $1 \le l \le o_i$, and to joining every $\bar{x}_{ik}$ to v_{ik}, $1 \le k \le \bar{o}_i$. Setting x_i = false will correspond to joining every x_{il} to $\bar{u}_{il}$, and to joining every $\bar{x}_{ik}$ to $\bar{v}_{ik}$.

For every entry terminal x_{il} (resp. $\bar{x}_{il}$) let the set of exit terminals, called the choice set, be $X_{il} = \{u_{il}, \bar{u}_{il}\}$ (resp. $\bar{X}_{il} = \{v_{il}, \bar{v}_{il}\}$). Then the clause

$(x_{il_1} \vee x_{jl_2} \vee x_{kl_3})$ is satisfied if and only if the density created by the 3 nets corresponding to the x's in the clause is less than 3. The y's (resp. $\bar{y}$'s) are used to make the assignment consistent; i.e., if x_i is assigned a true-value in one clause, it has to be assigned a true-value in all the other clauses in which it appears. Let the n leftmost entry terminals be $x_{i0},\ldots,x_{n0}$, and let the n rightmost exit terminals be $v_{1,\bar{o}_1+1}, \cdots, v_{n,\bar{o}_n+1}$. Let the choice sets for the entry terminals be as follows:

choice set for	x_{i0} is	$X_{i0} = \{\bar{u}_{i1}, v_{i1}\}$	
	x_{il}	$X_{il} = \{u_{il}, \bar{u}_{il}\}$	$1 \le l \le o_i$
	y_{il}	$Y_{il} = \{u_{il}, \bar{u}_{i,l+1}\}$	$1 \le l < o_i$
	y_{i,o_i}	$Y_{i,o_i} = \{u_{i,o_i}, v_{i,\bar{o}_i+1}\}$	
	$\bar{x}_{il}$	$\bar{X}_{il} = \{\bar{v}_{il}, v_{il}\}$	$1 \le l \le \bar{o}_i$
	$\bar{y}_{il}$	$\bar{Y}_{il} = \{\bar{v}_{il}, v_{i,l+1}\}$	$1 \le l \le \bar{o}_i$

This construction links all occurrences of x_{il} and $\bar{x}_{ik}$ together, $1 \le l \le o_i$, $1 \le k \le \bar{o}_i$, so that either every x_{il} is assigned to u_{il} and every $\bar{x}_{ik}$ is assigned to v_{ik}, or every x_{il} is assigned to $\bar{u}_{il}$ and every $\bar{x}_{ik}$ is assigned to $\bar{v}_{ik}$. Figure 5.4 shows these two possible assignments. It is clear that, given a boolean formula, the 2-choice problem corresponding to the construction described above can be created in polynomial time.

We claim that a given boolean formula is satisfiable if and only if the corresponding 2-choice problem has a solution of density less than $n+6$. Assume that the boolean formula is satisfiable. Assign every x_{il} to u_{il} and every $\bar{x}_{ik}$ to v_{ik} if x_i is true in the formula. Assign every x_{il} to $\bar{u}_{il}$ and every $\bar{x}_{ik}$ to $\bar{v}_{ik}$ if x_i is false. In each case there remains only one exit terminal to be assigned to each of the y's and $\bar{y}$'s. The construction created for each variable x_i adds 1 to the density in the columns between position n and $n+6(k_1+k_2)$.

Since at least one x_{il} (resp. $\bar{x}_{ik}$) in each clause is assigned to the exit terminal at the same position (otherwise the formula is not satisfied), the density of the solution to the 2-choice problem obtained in this way is at most $n+4$.

Now assume that the 2-choice problem has a solution with a density less that $n+6$. Observe that as soon as one x_{il}, y_{il}, $\bar{x}_{ik}$, or $\bar{y}_{ik}$ chooses its exit terminal, the exit terminals to be chosen by the remaining entry terminals with i as first subscript have been determined. This corresponds to a consistent true/false assignment to x_i in the boolean formula. Furthermore, setting every x_i to true if x_{i1} is joined to u_{i1} and to false if x_{i1} is joined to $\bar{u}_{i1}$ results in a true/false assignment to the n variables that satisfies the boolean formula. This is seen as follows. If one positive clause is not true, then the 3 exit terminals corresponding to the x's in this clause are assigned to the $\bar{u}$'s, which then implies that the 3 y's are assigned to the u's. Hence, density $n+6$ is achieved in column x_{kl_3}, a contradiction. A similar argument holds when a negative clause is not true. This concludes the proof of the Theorem. □

6. The One-Sided Range Problem

We next consider the version of the range problem in which every $l_i=1$. We show how to solve this *one-sided range problem* in $O(n\log n)$ time when every position on the lower row of the channel contains an exit terminal; i.e., $q_i=i$ for $1\le i\le M$. The input now consists of a list $L1$ containing n entry terminals and their right bounds $(p_1,r_1),\cdots,(p_n,r_n)$ with $1\le p_1<\cdots<p_n\le M$ and $1\le r_i\le M$. The solution is a CRP of minimum density consisting of the nets $(p_i,q_{f(i)})$ with $1\le q_{f(i)}\le r_i$. See Figure 6.0(a) and (d). In this section we call a net $(p_i,q_{f(i)})$ a right net if $p_i<q_{f(i)}$, a left net if $p_i>q_{f(i)}$, and a trivial net if $p_i=q_{f(i)}$. We call an entry (p_i,r_i) with $p_i\le r_i$ a *right entry* (since the net $(p_i,q_{f(i)})$ can be a right net), and we call an entry (p_i,r_i) with $p_i>r_i$ a *left entry* (since the net $(p_i,q_{f(i)})$ has to be a left net).

The technique used to determine the optimal assignment in the one-sided range problem is somewhat different from the one used in the previous sections. Instead of computing the optimal density and then using an assignment algorithm, we describe an algorithm that achieves the optimal density in time $O(n \log n)$ without pre-computing. Our algorithm is preceded by two preprocessing steps. In the first one we generate a list $L2$ containing the entries $(r_{i_1}, p_{i_1}), \ldots, (r_{i_n}, p_{i_n})$ sorted lexicographically in increasing order. We also determine the position of each p_i in $L2$ and the position of each r_{i_j} in $L1$. This information can be generated in $O(n \log n)$ time.

In the second preprocessing step we transform the problem into an equivalent one with distinct r_i's. The tie-breaking rules for making the r_i's distinct are as follows. If a left and a right entry have the same r_i, we change the right bound of the right entry to $r_i - 1$. If two left entries, respectively two right entries, have the same r_i, we change the right bound of the entry with the smaller p_i to $r_i - 1$. See Figure 6.0(b). The justification for these rules is as in Section 4.

We make the r_i's distinct by scanning list $L2$ from right to left. We apply the tie-breaking rule at all positions in the channel that contain two or more right bounds initially, and at positions that received two or more right bounds as a result of applying the tie-breaking rule to positions to the right. We use a heap to store the entries encountered during the scan of $L2$ and whose right bounds have not yet been changed to their final value. Assume that we are at position x, and that there are k entries with $r_i = x$. Insert the k entries in the heap, and delete the entry with the largest p_i value. The final right bound of the deleted entry is x. (Note that we did not need a heap in Section 4 because the l_i's and r_i's were ordered.) It can easily be shown that this procedure changes the right ranges correctly and that it can be implemented in $O(n \log n)$ time. We thus have the following Lemma.

Lemma 6.1 Every one-sided range problem can be transformed in $O(n \log n)$ time into an equivalent problem in which all r_i's are distinct.

Proof: Follows from the above discussion. □

From now on we assume that all the r_i's are distinct. Let $inr(x)$ be the number of left entries (p_i, r_i) with $p_i > x$ and $r_i \leq x$. These entries need an exit terminal $q_{f(i)} \leq x$ and thus cross column x coming from the right. Since $inr(x)$ exit terminals are needed by entries passing through column x from the right, right entries with $p_i \leq x$ and $r_i > x$ may have to get assigned to an exit terminal $> x$ and thus pass though column x coming from the left. As an example, see the entry (p_2, r_2) in Figure 6.0(a). Let $inl(x)$ be the number of right entries that have to pass through column x from the left. If we define $\sigma(x)$ to be equal to x if $x \geq 0$ and to equal 0 otherwise, then $inl(x) = \sigma(inr(x) + up(1,x) - x)$. Recall that now $down(1,x)=x$. We now define a quantity Γ, which we will show to be the best achievable density.

Definition 6.2: Let Γ be defined as follows:

$$\Gamma = \max_x \{ inr(x) + inl(x) \}$$

Lemma 6.3 $\hat{d} \geq \Gamma$.

Proof: The proof is along the lines of that of Lemma 4.2 and is omitted. □

First, observe that there exists an optimal solution in which every right entry (p_i, r_i) is assigned to an exit terminal at a position $\geq p_i$ (since all the r_i's are distinct). A right entry (p_i, r_i) that is assigned to the exit terminal at position p_i does not add to the density. Hence, each right entry initially claims the exit terminal at p_i as the one it is assigned to. However, we may not be able to assign every right entry to the exit terminal in position p_i. If a left entry (p_j, r_j) does not find an unclaimed exit terminal (i.e., an exit terminal not claimed by a right entry) at a position $\leq r_j$, a right entry (p_i, r_i) with $p_i \leq r_j$ has to give up its claim on the exit terminal at position p_i and take one

at a position $> p_i$. Hence, a right entry will be assigned to an exit terminal so that the resulting net is either a trivial or a right net, but never a left net. This leads to an algorithm for the one-sided range problem that consists of 4 phases. Each phase can be implemented in $O(n)$ time.

Phase 1. Tentative trivial net assignment: all right entries (p_i, r_i) claim the exit terminal p_i.

Phase 2. Determine the right entries whose claimed exit terminals are needed for left entries.

Phase 3. Assignment of the left entries.

Phase 4. Assignment of the remaining right entries.

Phase 1 can easily be implemented in $O(n)$ time. In Phase 2 we scan list $L2$ from left to right. Assume we determine at position x that an additional exit terminal at a position $\leq x$ is needed. Then the rightmost right entry (p_i, r_i) with $p_i \leq x$ and $r_i > x$ that was tentatively assigned as a trivial net gives up its claim on the exit terminal at position p_i. We say that the right entry (p_i, r_i) is *taken out*. Note that the taken out right entry (p_i, r_i) will always encounter a free exit terminal between position $x+1$ and r_i. This follows from the fact that the r_i's are distinct. In order to implement Phase 2 in $O(n)$ time we use an auxiliary list that allows us to find the next right entry to be taken out in constant time. The auxiliary list contains initially all right entries of list $L2$, and we delete entries that are taken out or that have $r_i \leq x$. Note that for achieving density Γ it is not crucial that the right entry closest to column x is taken out. This is a convention used by the algorithm that enables us to determine the entry that can be taken out efficiently.

In Phase 3 we scan the list $L2$ from right to left and assign each left entry (p_i, r_i) to the first exit terminal $x \leq r_i$ that is not used in a trivial net. The left entry (p_i, r_i) has precedence over other left entries (p_j, r_j) with $r_j < r_i$. We record in list $L1$ and $L2$ the exit terminal assigned to each left entry. Thus, Phase 3 can be implemented in $O(n)$ time.

In Phase 4 we assign the remaining right entries. Recall that each taken out right entry (p_i, r_i) gets assigned to an exit terminal at a position $> p_i$. We scan list $L2$ from right to left. A list of length $O(n)$ containing information about the positions of the free exit terminals is initially obtained from $L2$, and is scanned simultaneously. Assume we are at position x where $x = r_i$ and (p_i, r_i) is a taken out right entry. We then assign p_i to the rightmost free exit terminal at a position $\leq x$. This exit terminal can be determined in $O(1)$ time. It is possible that two taken out right entries are assigned to exit terminals so that the corresponding nets cross each other when they don't have to. But the contribution to the density of two crossing right nets is identical to the contribution to the density of the two uncrossed nets. The list containing the information about the free exit terminals can be handled so that the total amount of time needed to assign the taken out right nets in $O(n)$.

Theorem 6.4 The one-sided range problem can be solved in time $O(n \log n)$ by an algorithm that achieves density Γ.

Proof: The correctness and the $O(n \log n)$ time bound follow from the above discussion. Note that only the preprocessing steps require $O(n \log n)$ time. It remains to show that the algorithm achieves density Γ. Figure 6.1 shows the 10 configurations that can occur at any position x, $1 \leq x \leq M$. In the CRPs generated by the algorithm configurations 6.1(e) and (i) cannot occur since a right entry is only taken out when its exit terminal is needed by a left entry.

Let d_x be the density in column x. We will now show that for every x, $d_x \leq \Gamma$. Suppose there exists a column x with $d_x > \Gamma$. Choose x so that column $x+1$ has a smaller density; i.e., $d_{x+1} < d_x$. The density in column x is created by dr_x right nets and dl_x left nets where $dr_x + dl_x = d_x$. From the algorithm it follows that $dr_x \leq dl_x$. In column $x+1$ the situation 6.1(b), (c), or (f) has to occur. (All other situations are density increasing or density preserving.) The exit terminal at position $x+1$ is taken by a right net in

situation 6.1(b) and (f) and is empty in situation 6.1(c). In both cases no left entry could take this exit terminal in Phase 3, and hence, $dl_x = inr(x)$. If $dr_x = 0$, then $inr(x) > \Gamma$, a contradiction. If $dr_x > 0$, the dr_x right entries crossing column x have been forced out by left entries. The number of forced out right entries is $inr(x) + up(1,x) - x$. Hence, $inr(x) + inl(x) = dl_x + dr_x > \Gamma$, a contradiction, and the Theorem follows. □

7. Choosing Fewer Than n Nets

Consider the one-to-any problem with the following modification: Instead of choosing n nets, we want to choose s nets, $s < n$, such that the resulting CRP has minimum density. In other words, we must select s of the n entry terminals, s of the m exit terminals, and then assign the j^{th} selected entry terminal to the j^{th} selected exit terminal, $1 \le j \le s$. We show that this problem can be solved in time $O((n+m)\log\hat{d})$, where $\hat{d}$ is the optimal density. First we give a verification algorithm which, given an integer k, decides in time $O(m+n)$ whether $\hat{d} \le k$. The claimed result follows by using such a verification algorithm to compute $\hat{d}$ in a binary search manner.

The algorithm consists of a scan of the list of entry terminals and that of exit terminals. Let ENT (resp. EX) be the index of the entry (resp. exit) terminal currently scanned (i.e. if the scan is at p_i and q_j then $ENT = i$ and $EX = j$). Let $COUNT$ contain the number of nets created so far.

Outline of the Verification Algorithm

Input: The p_i's, the q_i's, s and an integer k.

Output: 'Yes' if $\hat{d} \le k$, 'No' otherwise.

Step 0: Set $ENT \leftarrow 1$, $EX \leftarrow 1$, $COUNT \leftarrow 0$.

Step 1: Repeat Step 2 until $COUNT = s$ or $EX = m+1$ or $ENT = n+1$.

Step 2 If joining p_{ENT} to q_{EX} does not cause the density to exceed k then create the net (p_{ENT}, q_{EX}) and increment each of ENT, EX and $COUNT$ by one. Otherwise do the following:

Case 1: If $p_{ENT} < q_{EX}$ then increment *ENT* by one.
Case 2: If $q_{EX} < p_{ENT}$ then increment *EX* by one.

Step 3: If $COUNT = s$ then output 'Yes', otherwise output 'No'.
End of Verification Algorithm

To prove correctness of the above algorithm, it suffices to show that there is a solution with density $\leq k$ if and only if the above algorithm answers 'Yes'. If the above algorithm answers 'Yes' then it is clear that the s nets it produced are a solution whose density is $\leq k$. We now show that if there is a solution whose density is $\leq k$ then our algorithm answers 'Yes'. Let $\alpha_1,...,\alpha_l$ be the nets produced by our verification algorithm. To prove that our algorithm answers 'Yes' it suffices to show that $l = s$. Among all valid solutions (i.e. solutions with density $\leq k$), choose the one having as many nets as possible in common with the nets $\alpha_1,...,\alpha_l$. Let $\beta_1,...,\beta_s$ be the nets of this valid solution. We claim that $\alpha_i = \beta_i$ for every $1 \leq i \leq l$. We prove this claim by induction on i:

Basis: First, note that $\alpha_1 = (p_1, q_1)$. If $\beta_1 \neq \alpha_1$ then the nets $\alpha_1, \beta_2, ..., \beta_s$ constitute another valid solution which has one additional net in common with $\alpha_1,...,\alpha_l$, a contradiction. Hence, $\beta_1 = \alpha_1$.

Induction: Suppose that $\alpha_j = \beta_j$ for all $j < i$ and that $\alpha_i \neq \beta_i$. To get a contradiction, it suffices to show that setting β_i equal to α_i yields a valid solution. Let $\alpha_i = (a, b)$ and $\beta_i = (c, d)$. From Step 2 of the algorithm, it follows that $a \leq c$ and $b \leq d$. This implies that if we change β_i from (c, d) to (a, b), $\beta_1,...,\beta_s$ still form a valid solution, a contradiction.

This completes the proof of the claim. The claim implies that $l = s$, because if $l < s$ then the algorithm would have selected at least one additional net α_{l+1} (possibly $\alpha_{l+1} = \beta_{l+1}$). This completes the correctness proof of the verification algorithm.

Theorem 7.1 The modified one-to-any problem in which only s nets are generated, $s < n$, can be solved in time $O((m+n)\log\hat{d})$.

Proof: It is not hard to see that the verification algorithm can be implemented to run in time $O(m+n)$. Since $\log\hat{d}$ applications of that algorithm produce an optimal solution, it follows that such an optimal solution can be found in time $O((m+n)\log\hat{d})$. □

The above modification for choosing only s nets can also be applied to the one-to-any problem with fixed nets and to the range problem with/without fixed nets. The details are left to the reader.

8. Conclusions

In this paper we presented efficient solutions to a number of different terminal assignment problems, and we showed that some more general versions of the problems considered are NP-hard. We introduced a new way to characterize the optimal density for the one-to-any and the ordered range problem both with and without fixed nets. Once the best achievable density was known, our assignment strategy produced an optimal assignment of entry to exit terminals. For two other assignment problems different methods were used to obtain the optimal assignment efficiently.

References

[AH] M.J. Atallah, S.E. Hambrusch, 'Optimal Rotation Problems in Channel Routing', Techn. Report, CSD-TR-467, Purdue University, 1984.

[AHU] A.V. Aho, J.E. Hopcroft, J.D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, 1974.

[BB] T. Bolognesi, D.J. Brown, 'A Channel Routing Algorithm with Bounded Wire Length', unpublished manuscript, 1982.

[DKSSU] D. Dolev, K. Karplus, A. Siegel, A. Strong, J.D. Ullman, 'Optimal Wiring between Rectangles', *Proceedings of the 13th Annual ACM Symp. on Theory of Computing*, pp 312-317, 1981.

[GCW] I.S. Gopal, D. Coppersmith, C.K. Wong, 'Optimal Wiring of Movable Terminals', *IEEE Trans. on Computers*, Vol. c-32, 9, pp 845-858, 1983.

[GJ] M.R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman & Co., San Francisco, 1979.

[L] E. L. Lawler, *Combinatorial Optimization: Networks and Matroids*, New York, Holt, Rinehart & Winston, 1976.

[LL] D.T. Lee, J.Y.-T. Leung, 'On the 2-Dimensional Channel Assignment Problem'. *IEEE transactions on Cpmuters*, Vol. C-33, pp 1-6, 1984.

[LaP] A.S. LaPaugh, R.Y. Pinter, 'On Minimizing Channel Density by Lateral Shifting', *Proceedings of 1983 ICCAD Conference*, 1983.

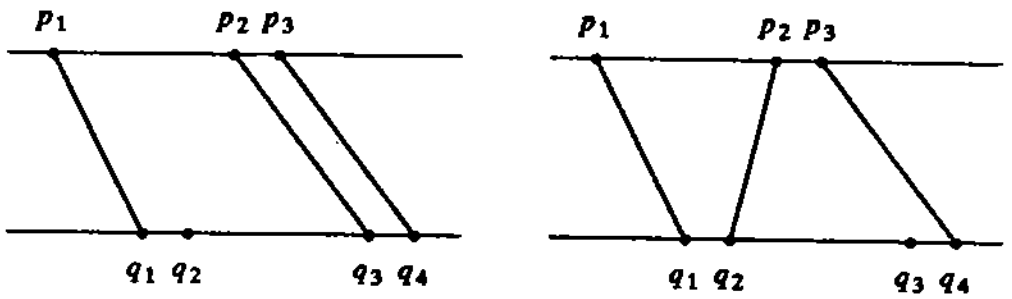
[LeP] C.E. Leiserson, R.Y. Pinter, 'Optimal Placement for River Routing', *SIAM Journal on Computing*, Vol. 12, Nr. 3, pp 447-462, 1983.

[M] A. Mirzaian, 'Channel Routing in VLSI', *Proceedings of 16th Annual ACM Symp. on Theory of Computing*, 1984.

[P] R.Y. Pinter, 'The Impact of Layer Assignment Methods on Layout Algorithms for Integrated Circuits', Ph.D. Thesis, MIT, 1982.

[PL] F.P. Preparata, W. Lipski, 'Three Layers are enough', *Proceedings of the 23rd Annual IEEE Foundations of Comp. Sc. Conf.*, pp 350-357, 1982.

[RBM] R.L. Rivest, A.E. Baratz, G. Miller, 'Provably Good Channel Routing Algorithms', *Proceedings of the CMU Conf. on VLSI Systems and Computations*, pp 153-159, 1981.



(a) non-optimal (b) optimal

Figure 2.0

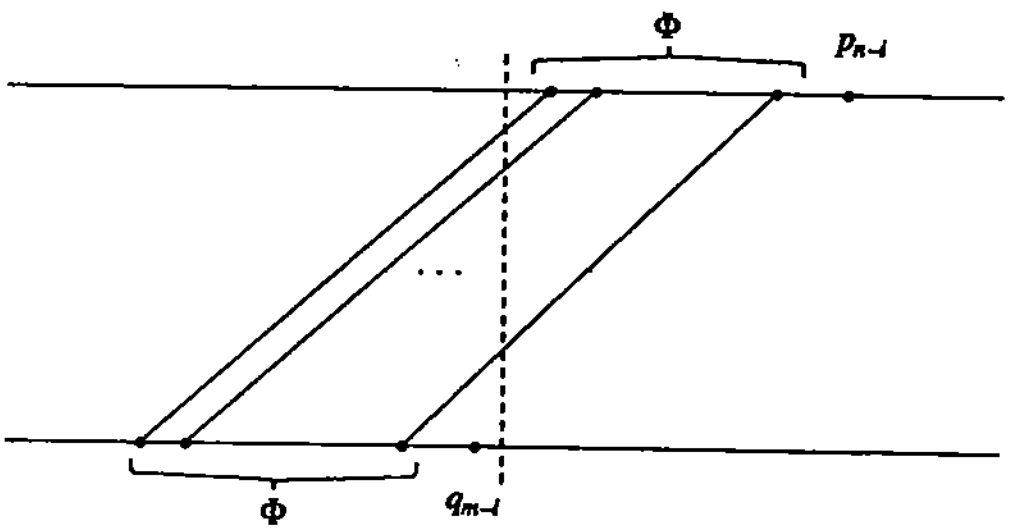
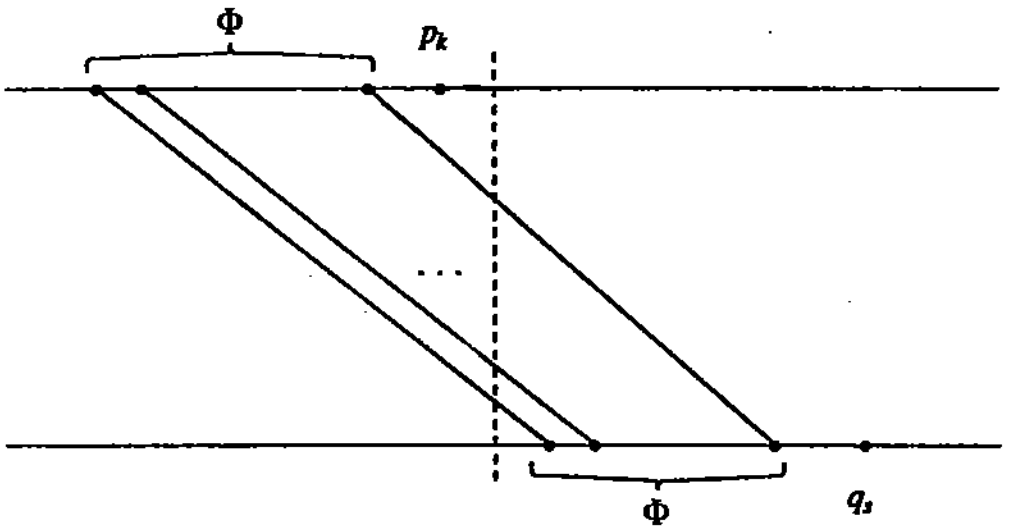
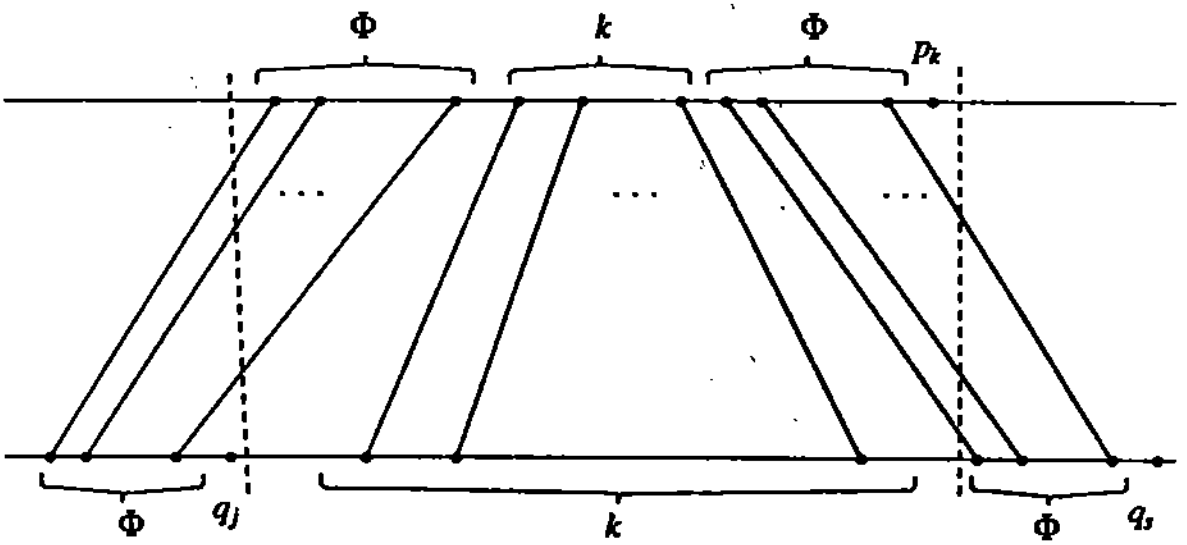


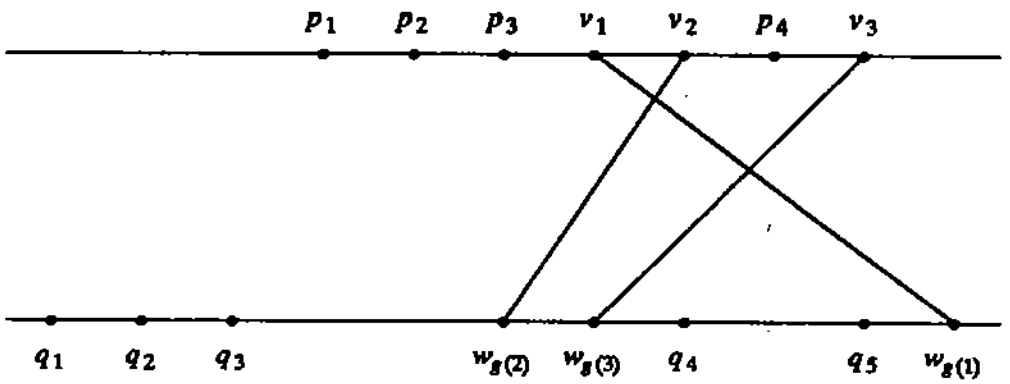
Figure 2.1



Situation B when there are no free exit terminals to the left of q_s
Figure 2.2



Situation B when there are free exit terminals to the left of q_s
Figure 2.3



One-to-any problem with 3 fixed nets
Figure 3.0

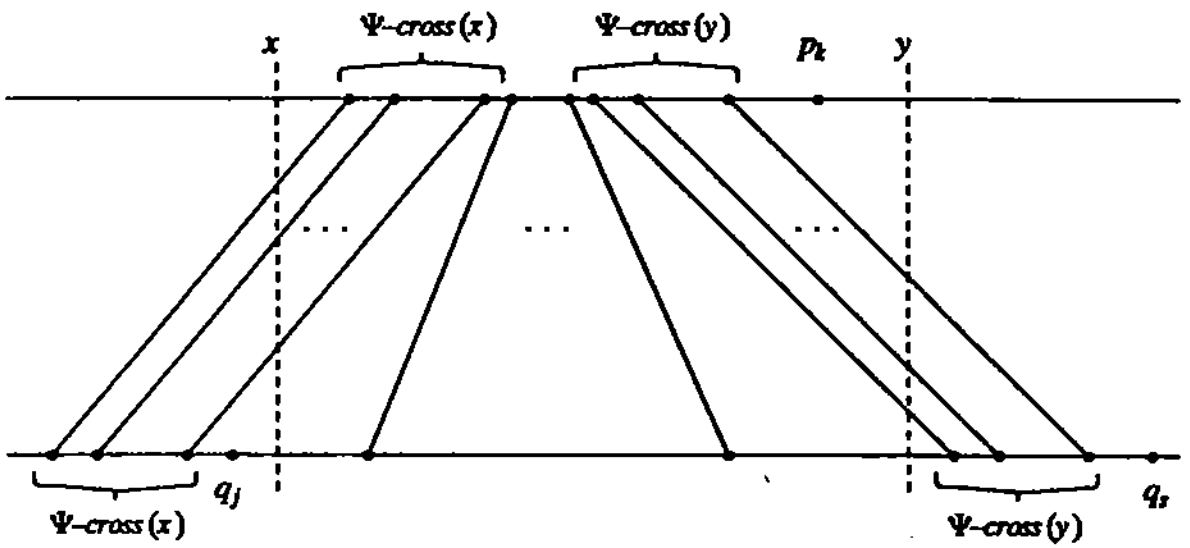


Figure 3.1

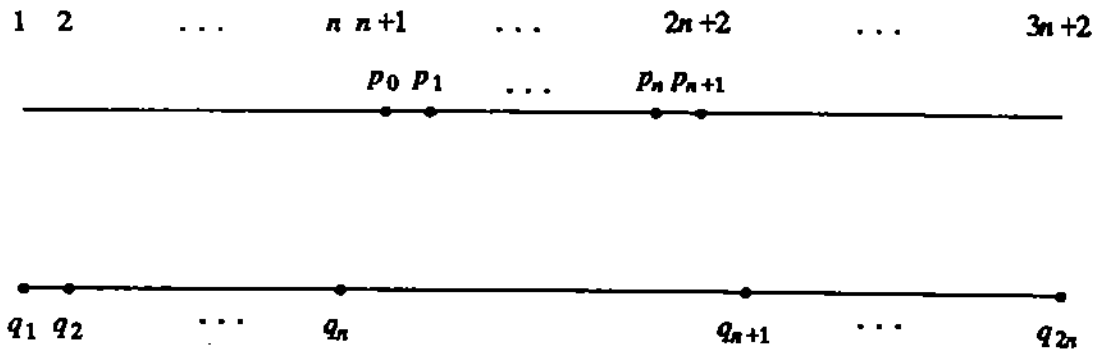
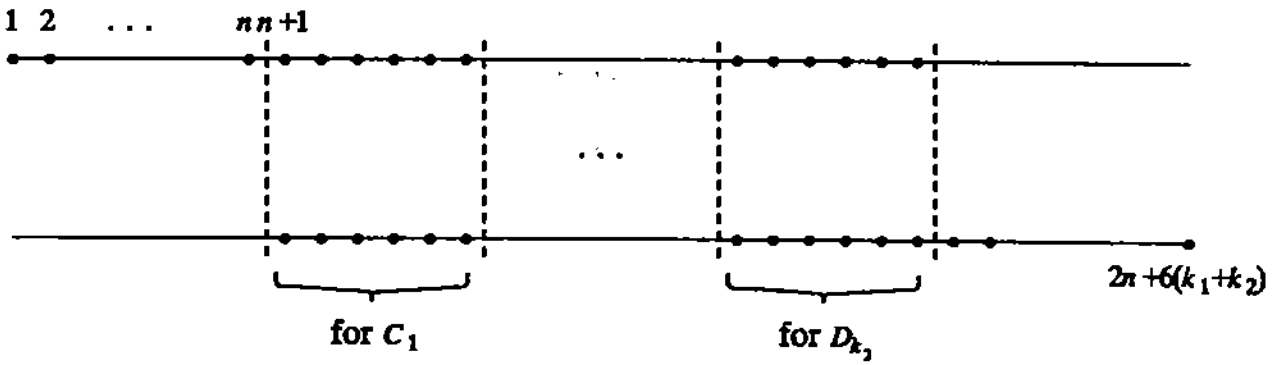
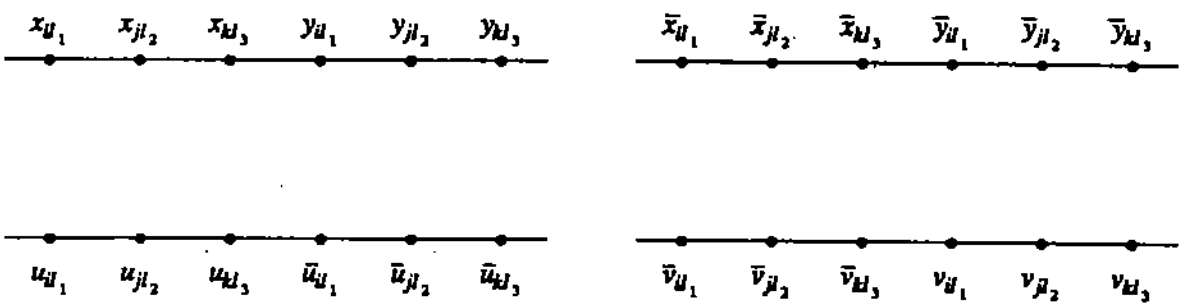


Figure 5.1

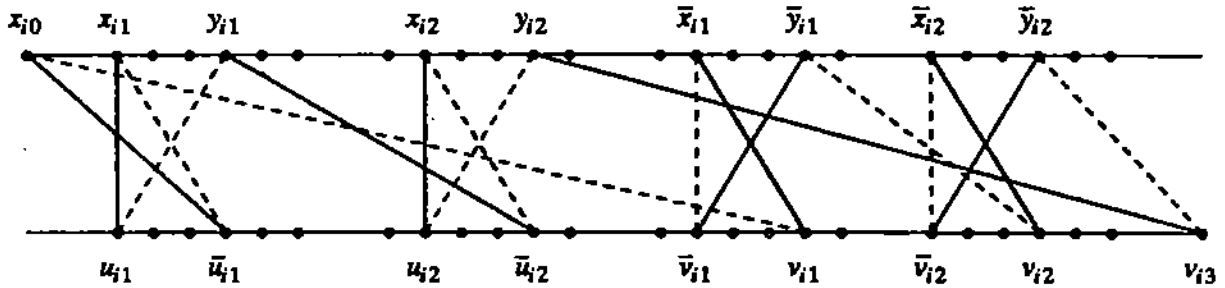


Organization of the channel
Figure 5.2



(a) Terminals for the clause $(x_{il_1} \vee x_{jl_2} \vee x_{kl_3})$ (b) Terminals for the clause $(\bar{x}_{il_1} \vee \bar{x}_{jl_2} \vee \bar{x}_{kl_3})$

Figure 5.3



Solid lines show the assignment corresponding to x_i = true,
dashed lines show the assignment corresponding to x_i = false
Figure 5.4

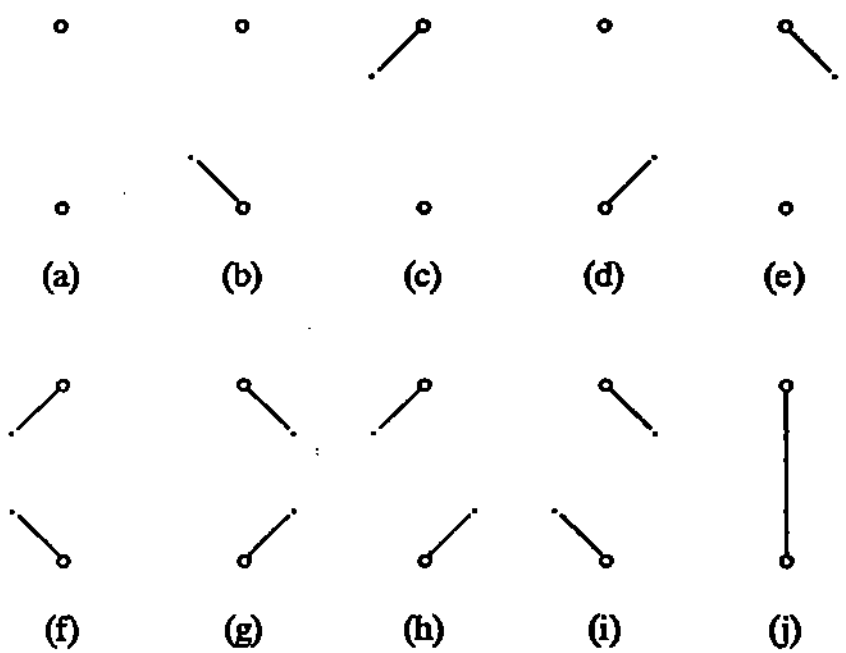
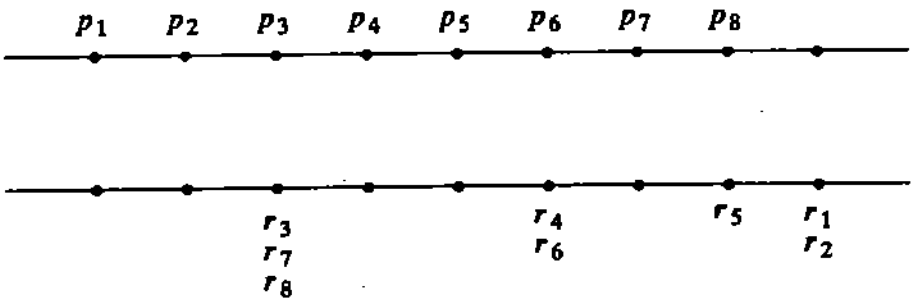
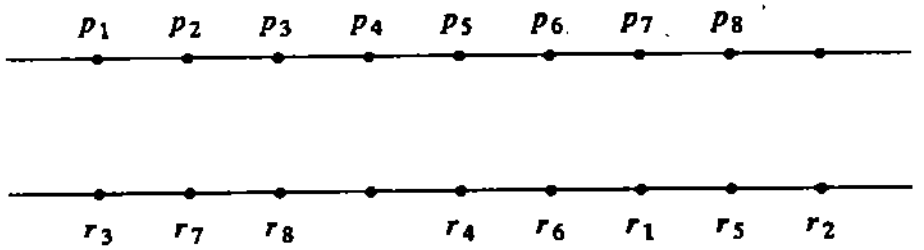


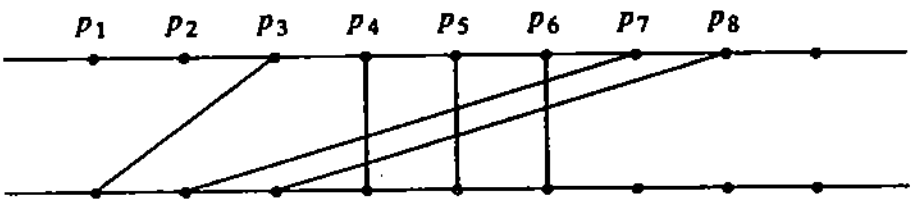
Figure 6.1



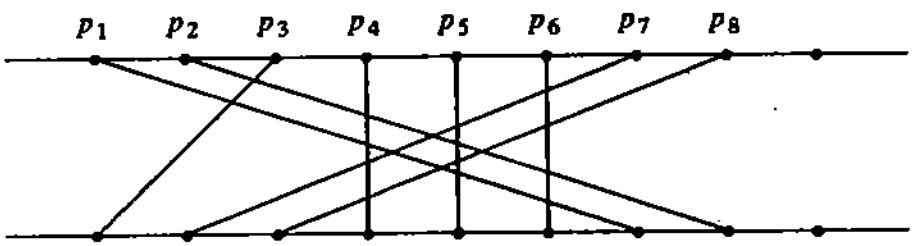
(a) initial input



(b) after preprocessing



(c) after phase 3



(d) final assignment
Figure 6.0